Chapter 05

基本輸出/輸入函式

學習綱要

- 5-1 printf()
- 5-2 scanf()
- 5-3 亂數的取得
- 5-4 聲音的產生

學習目標

- 1. 能夠以 printf() 輸出資料。
- 2. 能夠以 scanf() 輸入資料。
- 3. 能取以 rand() 取得亂數。
- 4. 能夠以 beep() 產生聲音。
- 5. 能實作基本輸出與輸入函式。

C 語言採用 printf() 與 scanf() 作爲基本輸出入函式。其次，本單元也要介紹亂數的取得，與聲音的產生，請看以下說明。

5-1 printf()

printf() 可輸出結果，其中括號內須放置一對雙引號『“”』來將欲輸出的資料當作引數傳給 printf() 函式。例如：

```
printf("a");
printf("aa");
```

的輸出結果是：

```
aaa
```

但若要換行與歸位，則應加上跳脫字元『\n』，例如：

```
printf("a\n");
printf("aa");
```

的輸出結果是：

```
a
aa
```

其次，如果是變數的輸出，則必須在輸出敘述中加入一個列印格式，此列印格式以控制字元『%』符號開頭，如表 5-1：

▶ 表 5-1 printf() 列印格式表

資料型別	列印格式
整數	%d
長整數	%ld
浮點數 float	%f
浮點數 double	%lf
字元	%c
字串	%s

然後編譯器會依照控制字元的順序，到後面引數區取對應資料來輸出。例如：

```
int a=3;
printf("%d", a);
```

『%』即表示要到後面引數區取變數來取代，『d』是將變數視爲整數，所以輸出結果是：

```
3
```

若是：

```
int a=3, b=5;
printf("%d  %d", a, b);
```

有兩個控制字元，那就到後面引數區依照順序取變數輸出，所以輸出是：

```
3 5
```

字串區也可以同時有一般字元與控制字元『%』，此時一般字元就直接輸出，控制字元就到後面引數區依序取變數輸出。例如：

```
int a=3, b=5;
printf("a=%d  b=%d", a, b);
```

因爲『"a= "』是一般字元，就直接輸出，『% d』是控制字元，就去引數區按照順序取對應變數來取代。第一個『%d』取到 a，第二個『%d』取到 b，所以輸出結果是：

```
a=3     b=5
```

輸出過程中如須換行與歸位，也是以跳脫字元序列『 \n 』來完成。例如：

```
int a=3, b=5;
printf("a=%d\n", a);
printf("b=%d", b);
```

的結果是：

```
a=3
b=5
```

自我練習

1. 請自行鍵入以下程式，並觀察執行結果。

```
int main() {
    int a=97;
    printf("%d\n", a);
    printf("%c\n", a);
    printf("%c\n", (char)a);
    char b='a';
    printf("%c\n", b);
    printf("%d\n", b);
    printf("%d\n", (int)b);
    char c='ab';
    printf("%c\n", c);
    char d='abc';
    printf("%c\n", d);
    char e='a ';
    printf("%c\n", e);
    return 0;
}
```

資料的對齊

輸出資料時，亦可指定資料長度，那就可以將所有資料對齊。例如：

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int a=3,b=55;
    double c1=3.4,c2=15.678;
    printf("a =%8d,b =%8d\n",a,b);//佔8格，靠右
    printf("c1=%8.2f,c2=%8.2f",c1,c2);//共8格，靠右，小數點取2位
    //請留意遇到『%』就是控制字元，就到後面按順序取對應資料輸出
    //非控制字元，就直接輸出，例如『a=』，就直接輸出。
    return 0;
}
```

的執行結果如下圖，這樣就可以對齊所有資料。

```
a =       3,b =      55
c1=    3.40,c2=   15.68
--------------------------------
```

網路C語言使用手冊

網路 C 語言的使用手冊很多，但是，https://en.cppreference.com/w 是我目前找到最齊全的參考手冊，有 C 也有 C++ 如圖 5-1：

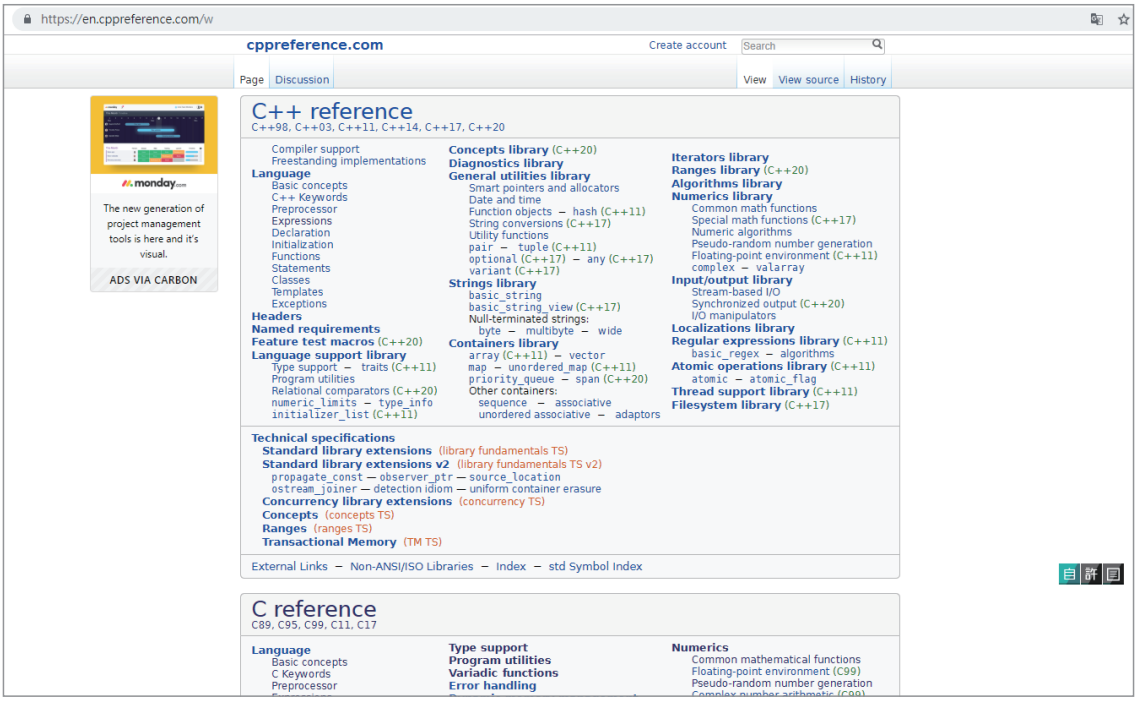
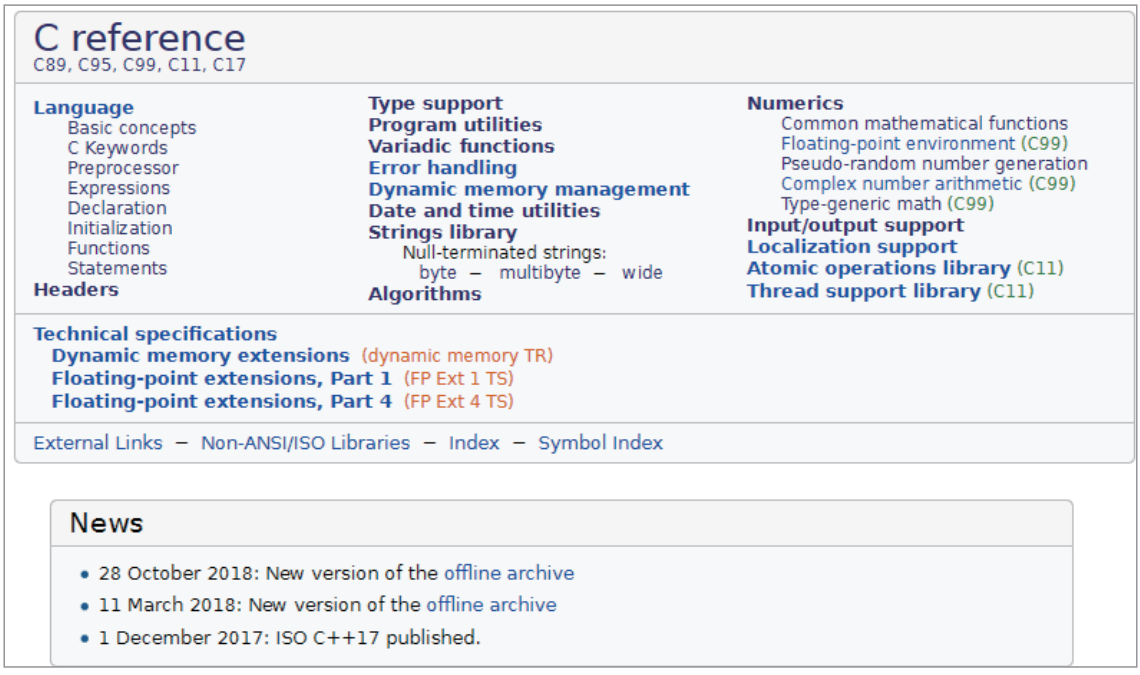


圖 5-1　C 語言網路使用手冊一

C 語言的詳細資料如圖 5-2（點選圖 5-1 的下面「C reference」）：上一章曾提到編譯器版本問題，請看下圖就會明瞭，若標示『C99』表示那是 1999 新增的指令；若標示『C11』表示那是 2011 新增的指令，沒標示則是 1990 制訂的標準。



C reference
C89, C95, C99, C11, C17

Language
Basic concepts
C Keywords
Preprocessor
Expressions
Declaration
Initialization
Functions
Statements
Headers

Type support
Program utilities
Variadic functions
Error handling
Dynamic memory management
Date and time utilities
Strings library
Null-terminated strings:
byte – multibyte – wide
Algorithms

Numerics
Common mathematical functions
Floating-point environment (C99)
Pseudo-random number generation
Complex number arithmetic (C99)
Type-generic math (C99)
Input/output support
Localization support
Atomic operations library (C11)
Thread support library (C11)

Technical specifications
Dynamic memory extensions (dynamic memory TR)
Floating-point extensions, Part 1 (FP Ext 1 TS)
Floating-point extensions, Part 4 (FP Ext 4 TS)

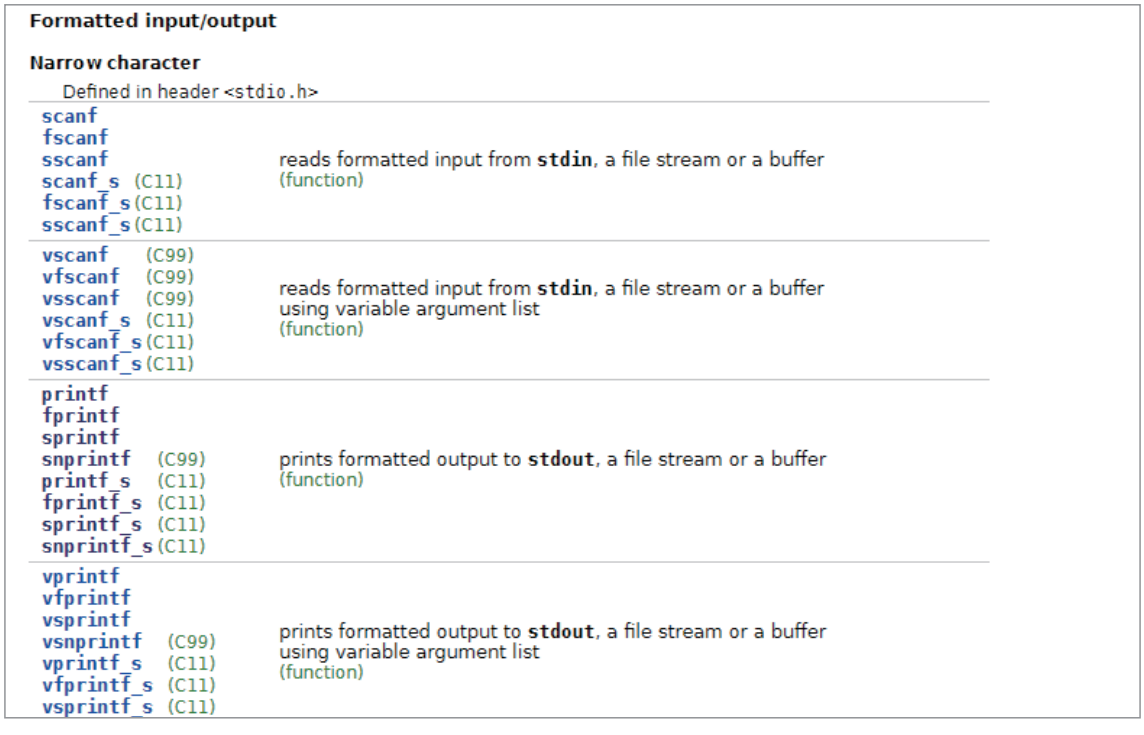
External Links – Non-ANSI/ISO Libraries – Index – Symbol Index

News

- 28 October 2018: New version of the offline archive
- 11 March 2018: New version of the offline archive
- 1 December 2017: ISO C++17 published.

圖 5-2　C 語言網路使用手冊二

於圖 5-2 點選『input/output support』，再往下捲動，即可找到 printf() 的詳細說明，如圖 5-3：



Formatted input/output

Narrow character

Defined in header <stdio.h>

scanf **fscanf** **sscanf** **scanf_s** (C11) **fscanf_s** (C11) **sscanf_s** (C11)	reads formatted input from **stdin**, a file stream or a buffer (function)
vscanf (C99) **vfscanf** (C99) **vsscanf** (C99) **vscanf_s** (C11) **vfscanf_s** (C11) **vsscanf_s** (C11)	reads formatted input from **stdin**, a file stream or a buffer using variable argument list (function)
printf **fprintf** **sprintf** **snprintf** (C99) **printf_s** (C11) **fprintf_s** (C11) **sprintf_s** (C11) **snprintf_s** (C11)	prints formatted output to **stdout**, a file stream or a buffer (function)
vprintf **vfprintf** **vsprintf** **vsnprintf** (C99) **vprintf_s** (C11) **vfprintf_s** (C11) **vsprintf_s** (C11)	prints formatted output to **stdout**, a file stream or a buffer using variable argument list (function)

圖 5-3　C 語言網路使用手冊三

於圖 5-3 點選『printf』，畫面如圖 5-4：這裡有 printf() 的詳細用法：

C / File input/output

printf, fprintf, sprintf, snprintf, printf_s, fprintf_s, sprintf_s, snprintf_s

Defined in header <stdio.h>

`int printf( const char *format, ... );` `int printf( const char *restrict format, ... );`	(1)	(until C99) (since C99)
`int fprintf( FILE *stream, const char *format, ... );` `int fprintf( FILE *restrict stream, const char *restrict format, ... );`	(2)	(until C99) (since C99)
`int sprintf( char *buffer, const char *format, ... );` `int sprintf( char *restrict buffer, const char *restrict format, ... );`	(3)	(until C99) (since C99)
`int snprintf( char *restrict buffer, size_t bufsz, const char *restrict format, ... );`	(4)	(since C99)
`int printf_s(const char *restrict format, ...);`	(5)	(since C11)
`int fprintf_s(FILE *restrict stream, const char *restrict format, ...);`	(6)	(since C11)
`int sprintf_s(char *restrict buffer, rsize_t bufsz, const char *restrict format, ...);`	(7)	(since C11)
`int snprintf_s(char *restrict buffer, rsize_t bufsz, const char *restrict format, ...);`	(8)	(since C11)

Loads the data from the given locations, converts them to character string equivalents and writes the results to a variety of sinks.

1) Writes the results to the output stream `stdout`.
2) Writes the results to the output stream `stream`.
3) Writes the results to a character string `buffer`. The behavior is undefined if the string to be written (plus the terminating null character) exceeds the size of the array pointed to by `buffer`.
4) Writes the results to a character string `buffer`. At most `bufsz - 1` characters are written. The resulting character string will be terminated with a null character, unless `bufsz` is zero. If `bufsz` is zero, nothing is written and `buffer` may be a null pointer, however the return value (number of bytes that would be written not including the null terminator) is still calculated and returned.

圖 5-4　C 語言網路使用手冊四

圖 5-4 就有如字典般的詳盡了，所有參數還有詳細解說，如圖 5-5：

Parameters

stream - output file stream to write to

buffer - pointer to a character string to write to

bufsz - up to bufsz - 1 characters may be written, plus the null terminator

format - pointer to a null-terminated multibyte string specifying how to interpret the data.

The format string consists of ordinary multibyte characters (except %), which are copied unchanged into the output stream, and conversion specifications. Each conversion specification has the following format:

- introductory % character
- (optional) one or more flags that modify the behavior of the conversion:
  - **-**: the result of the conversion is left-justified within the field (by default it is right-justified)
  - **+**: the sign of signed conversions is always prepended to the result of the conversion (by default the result is preceded by minus only when it is negative)
  - *space*: if the result of a signed conversion does not start with a sign character, or is empty, space is prepended to the result. It is ignored if + flag is present.
  - **#** : *alternative form* of the conversion is performed. See the table below for exact effects otherwise the behavior is undefined.
  - **0** : for integer and floating point number conversions, leading zeros are used to pad the field instead of *space* characters. For integer numbers it is ignored if the precision is explicitly specified. For other conversions using this flag results in undefined behavior. It is ignored if - flag is present.

圖 5-5　C 語言網路使用手冊五

還有範例與執行結果，如圖 5-6：

Example

Run this code

```
#include <stdio.h>

int main(void)
{
    printf("Strings:\n");
    const char* s = "Hello";
    printf("\t.%10s.\n\t.%-10s.\n\t.%*s.\n", s, s, 10, s);

    printf("Characters:\t%c %%\n", 65);

    printf("Integers\n");
    printf("Decimal:\t%i %d %.6i %i %.0i %+i %i\n", 1, 2, 3, 0, 0, 4, -4);
    printf("Hexadecimal:\t%x %x %X %#x\n", 5, 10, 10, 6);
    printf("Octal:\t%o %#o %#o\n", 10, 10, 4);

    printf("Floating point\n");
    printf("Rounding:\t%f %.0f %.32f\n", 1.5, 1.5, 1.3);
    printf("Padding:\t%05.2f %.2f %5.2f\n", 1.5, 1.5, 1.5);
    printf("Scientific:\t%E %e\n", 1.5, 1.5);
    printf("Hexadecimal:\t%a %A\n", 1.5, 1.5);
}
```

Output:

```
Strings:
    .     Hello.
    .Hello     .
    .     Hello.
Characters:     A %
Integers
Decimal:        1 2 000003 0  +4 -4
Hexadecimal:    5 a A 0x6
Octal:          12 012 04
Floating point
Rounding:       1.500000 2 1.30000000000000004440892098500626
Padding:        01.50 1.50  1.50
Scientific:     1.500000E+00 1.500000e+00
Hexadecimal:    0x1.8p+0 0X1.8P+0
```

圖 5-6　C 語言網路使用手冊六

自我練習

1. 請於 www.tutorialspoint.com 網站探索 C 語言線上說明。
2. 請於 www.programiz.com 網站探索 C 語言線上說明。
3. 請於 www.learn-c.org 網站探索 C 語言線上說明。

5-2　scanf()

scanf( ) 可輸入資料，其使用方式與輸出 printf() 是一樣的方式，二者均須使用到控制字元『%』格式來控制輸出入的資料。因此，輸入字元、整數、浮點數及字串等會有不同的處理方式，請看以下說明。

輸入字元

輸入字元的敘述如下：

```
char a;
scanf("%c", &a);//請留意是&
```

可將輸入的資料放入在某個變數所指定位址中，取址符號『&』用來取得欲放置輸入資料的變數位址。例如，以下程式可輸入一個字元。

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    char a;
    printf("Please press any char: ");
    scanf("%c", &a); //請留意是&
    printf("Your char is %c", a);
    return 0;
}
```

請留意輸入資料要有取址符號『＆』，漏掉了就是錯誤。

自我練習

1. 請鍵入以上程式，並觀察執行結果。
2. 請鍵入以下程式，並觀察執行結果。

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    char a,b;
    printf("Please press any char: ");
    scanf("%c", &a);
    printf("Your char is %c", a);
    printf("Please press any char: ");
    scanf("%c", &b);
    printf("Your char is %c", b);
    return 0;
}
```

以上程式，第二次輸入字元

```
scanf("%c", &b);
```

會因鍵盤緩衝問題（只要有先輸入東西，例如整數、實數都有此問題），而直接跳過，這時請將字元前放一個前導空白字元就可以。所以程式修正如下：

```
scanf(" %c", &b);//留意%c的前導空白
```

這在使用手冊有提到 https://en.cppreference.com/w/c，請讀者自行瀏覽。

輸入數值

常用的數值型態有 int、long 與 double 等，若要輸入這些型態的數值，則應使用不同的列印格式，例如，%d、%ld 與 %lf 等。請輸入以下程式，並觀察執行結果。

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int    a;
    long   b;
    double  c;
    printf("Enter an integer:     ");
    scanf("%d", &a); //請留意是&
    printf("Enter a long integer: ");
    scanf("%ld", &b); //請留意是&
    printf("Enter a float:        ");
    scanf("%lf", &c); //請留意是&
    printf("\n Your enter..... \n");
    printf("integer      a =%d  \n",  a);
    printf("long integer b =%ld \n", b);
    printf("float        c =%f", c);
    return 0;
}
```

請留意輸入資料要有取址符號『&』，漏掉了就是錯誤。以下可輸入兩個數值，輸入資料時，資料中間以逗號分隔。

```
int a,b,d;
printf("input a,b:");
d=scanf("%d,%d",&a,&b);//請留意中間是逗號
printf("a+b=%d\n",a+b);
printf("d=%d\n",d);    //2   (傳回輸入資料正確的個數)
```

以上執行結果如下：（請自行補程式的頭尾）

```
input a,b:3,4
a+b=7
d=2
```

以下程式可輸入兩筆資料，輸入資料時，資料中間以空白分隔。

```
int a,b,d;
printf("input a b:");
d=scanf("%d %d",&a,&b);//請留意中間是空白
printf("a+b=%d\n",a+b);
printf("d=%d\n",d);    //2   (傳回輸入資料正確的個數)
```

以上程式執行結果如下：（請自行補程式的頭尾）

```
input a b:3 4
a+b=7
d=2
```

其次，scanf() 還傳回正確的資料個數，所以以上程式輸入 2 筆資料，d 值均傳回 2。若輸入資料型態不對，例如，故意輸入 a 3，由於輸入資料型態錯誤，將傳回『0』，或『EOF』，如下圖：

```
input a b:a 3
a+b=16
d=0
```

輸入字串

大部分程式語言都有字串 string 型態。但是 C 語言爲了加速處理速度，並沒有字串型態，而是使用字元陣列。因此在輸入字串時，我們必須利用字元陣列來完成輸入。例如，以下程式可輸入一個字串。

```
char c[20];//字元陣列，請看第七章
printf("Please enter a string: ");
scanf("%s", &c);//
printf("Your enter is %s", c);
```

以上輸出結果如下圖：請自行鍵入程式，並觀察執行結果。

```
Please enter a string: Gwosheng
Your enter is Gwosheng
```

關於字串的使用，請看本書第 9-3 節。

自我練習

1. 請鍵入以上程式，並觀察執行結果。
2. 請於 https://en.cppreference.com/w 查詢『scanf()』的詳細用法。
3. 請寫一個程式，可以輸入姓名、性別（m 代表男性，f 代表女性），且可以輸出姓名與性別。例如:輸入「煖維 m」，輸出「您的名字是煖維，性別：m」。
4. 請寫一個程式，可以輸入姓名、國文成績、數學成績，且可以輸出以上資料。例如「煖維您的國文成績爲 80, 數學成績爲 60, 總分爲 140, 平均爲 70」。

5-3 亂數

我們寫程式時，有時需要一個亂數，例如樂透開獎，樸克牌發牌等都需要產生一個亂數。所以電腦內部就以記憶體儲存一個亂數表，以供使用者取用。取用亂數的函式是：

```
rand();
```

它將傳回一個 0 到 32767 的任意整數。所以我們宣告一個整數變數暫存，如以下程式：

```
int a;
a=rand();
```

其次，此函式放在 stdlib.h，所以還要在標頭檔宣告如下：

```
#include <stdlib.h>
```

還有，因爲亂數表是固定的，那要如何取到不同亂數呢？答案是，可取系統時間當依據。因爲每次程式執行時，系統的時間一定不一樣，就可取到不同的亂數。以下敘述，以當時的系統時間，當作亂數表的起點。

```
srand(time(0));//以當時的系統時間，當作亂數表的起點
```

其次，time() 函式放在 time.h，所以也要載入以下標頭檔。

```
#include <time.h>//本例使用Time()函式
```

範例5-3a

示範以上亂數的取得。

程式列印

```
#include <stdio.h>
#include <stdlib.h>
//步驟1，引入標頭檔
#include <time.h>//本例使用time()函式
int main() {
    int a;
    //步驟2，以當時的系統時間，當作亂數表的起點
    srand(time(0));
    //步驟3，取亂數
    a=rand();
    printf("The Random Number is %d .\n", a);
    return 0;
}
```

以下敘述，可取 0~5 的亂數。「%」爲取餘運算子，將於 6-1 節介紹。

```
int a=rand()%6;
```

以下敘述，可取 1~6 的亂數。

```
int a=1+rand()%6;
```

以下敘述，可取 97~122 的亂數。而 97~122 爲小寫字元，所以可輸出小寫字元。

```
int a=97+rand()%26;
printf("%c",a);
```

自我練習

1. 請將以上程式的『srand(time(0));』去掉，並觀察執行結果，是否每次都一樣。
2. 請練習於 https://en.cppreference.com/w 尋找『Pseudo-random number generation』，如圖 5-7：

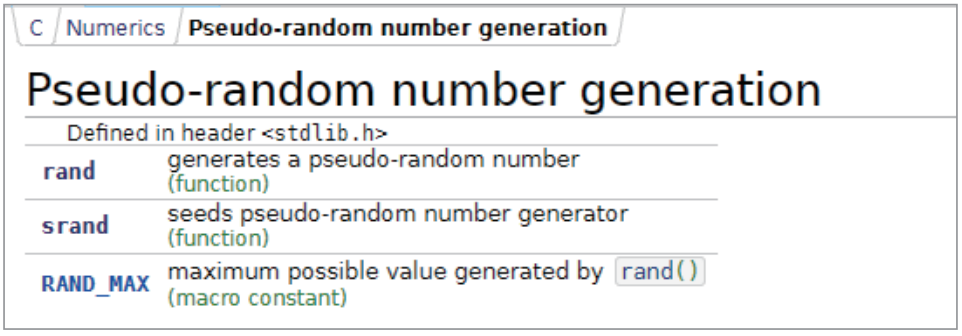


圖 5-7　random 說明

3. 請練習取 -5 到 8 的整數亂數，且輸出此數字。請參考 6-1 節取餘運算子『%』。
4. 請練習取 -1 到 1 的實數亂數，且輸出此數字。請參考 6-1 節取餘運算子『%』與除法『/』。
5. 請練習輸出任一大寫字元，且可由使用者也輸入此字元。

6. 若函式 rand() 的回傳值爲一介於 0 和 10000 之間的亂數，下列那個運算式可產生介於 100 和 1000 之間的任意數 (包含 100 和 1000) ？ (APCS10603)

(A) rand() % 900 + 100

(B) rand() % 1000 + 1

(C) rand()% 899 + 101

(D) rand()% 901 + 100

5-4　聲音的產生

要讓電腦發出聲音，可使用 _beep() 函式，此函式放在 stdlib.h，所以要載入標頭檔 #include <stdlib.h>，其語法如下：

```
void _beep(unsigned int x,unsigned int y);
```

x 是要發出的頻率，單位是 Hz，y 是發聲的時間，單位是 ms。其次，鋼琴鍵盤頻率對照表如表 5-2：

▶ 表 5-2　音階頻率表

	n	1	2	3	4	5	6	7	8	9	10	11	12
音階	音符	(Do)	(Do#)	(Re)	(Re#)	(Mi)	(Fa)	(Fa#)	(So)	(So#)	(La)	(La#)	(Si)
低音	頻率 (Hz)	262	277	294	311	330	349	370	392	415	440	466	494
中音	頻率 (Hz)	523	554	587	622	659	698	740	784	831	880	932	988
高音	頻率 (Hz)	1046	1109	1175	1245	1318	1397	1480	1568	1661	1760	1865	1976

例如，以下敘述可發出『Do』的音，且持續 0.5 秒。

```
_beep(523,500);
```

範例5-4a

示範發出 Do、Re、Me、Fa…等八個音。

程式列印

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    _beep(523,500);
    _beep(587,500);
    _beep(659,500);
    _beep(698,500);
    _beep(784,500);
    _beep(880,500);
    _beep(988,500);
    _beep(1046,500);
    return 0;
}
```

演奏音樂

要讓電腦演奏音樂，那就要先下載一個簡譜，如圖 5-8。

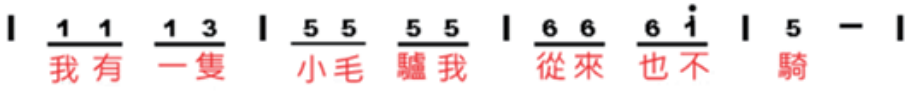


圖 5-8　小毛驢簡譜

本例一分鐘 80 拍，所以 1 拍的時間是，60000ms/80，以一個四分音符爲 1 拍，所以上圖一個八分音符所佔的時間是 60000ms/(2*80)，每小節是 2 拍。其次，上圖，所有音符的最小時間是八分音符，所以我就取八分音符的時間爲 1 個單位。第 1 音『1 我』半拍，取 1 個單位；『5- 騎』是 2 拍，取 4 個單位。

範例5-4b

示範演奏音樂。本範例僅作 4 小節，『我有一隻小毛驢我從來也不騎』。

程式列印

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int t=60000/160;
    _beep(523,t);
    _beep(523,t);
    _beep(523,t);
    _beep(659,t);
    _beep(784,t);
    _beep(784,t);
    _beep(784,t);
    _beep(784,t);
    _beep(880,t);
    _beep(880,t);
    _beep(880,t);
    _beep(1064,t);
    _beep(784,4*t);
    return 0;
}
```

補充說明

1. 請熟悉以上發音方式，第七章還會介紹如何製作電子琴。
2. 以上僅輸出一小節，程式就有點冗長，那寫完一首歌不就累垮。請放心，待介紹迴圈與陣列就會明瞭，本書於範例 9-5g，將會以陣列與迴圈重做本範例。

自我練習

1. 請繼續完成以上音樂。
2. 請用輸出指令讓輸出音樂的同時，營幕也輸出歌詞，那就是 KTV 了。
3. 請自己找一首歌，完成輸出歌詞與音階。

警車聲音

警車的警報聲是連續發出頻率 265Hz 與 350Hz 所形成，以下範例示範發出此聲音。

範例5-4c

示範發出警車警報聲。

程式列印

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    while(1){//無窮迴圈，請看第六章
        _beep(265,300);
        _beep(350,700);
        _sleep(50);//空白
    }
    return 0;
}
```

自我練習

1. 市內電話來電大都採用 1000Hz 與 500Hz 交錯形成的樂音，請寫程式完成。
2. 同上題，練習讓鈴聲可越來越急。(此題可待學完迴圈再作)
3. 音感練習。請觀察您家微波爐、洗衣機、汽車安全帶未繫、倒車警告、手機的各種聲響，並寫程式完成。(提示:若沒音感，可上網搜尋)